Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Audit&SécuRés

Projet

Communication IP par utilisation de canal caché sécurisé

Présentation du projet

Le but du projet est de réaliser un outil permettant de communiquer entre deux machines connectées par Internet au travers d'un « *canal caché* », c-à-d de manière invisible aux outils de surveillance pouvant intercepter les paquets IP utilisés pour cette communication.

Une présentation de l'ICANN sur les « canaux cachés *» peut être consultée à :*

`https://www.icann.org/fr/blogs/details/what-is-an-internet-covert-channel-29-8-2016-fr`

Les **contraintes** sont les suivantes :

- ▷ le **canal caché** doit se comporter comme un « *tunnel* » :
  - ◇ la communication entre les deux machines doit être possible en utilisant la couche transport de TCP/IP ;
  - ◇ les utilisateurs doit pouvoir utiliser un outil de communication normal, comme l'outil `socat`, pour établir ou attendre un échange basé TCP ou UDP ;
  - ◇ les datagrammes relatifs à ces communications doivent être transportés par le canal caché entre les deux machines ;

  ⟹ *On utilisera des interfaces Linux de type «* veth *» à deux interfaces liées :*
  - ⋆ *si on **injecte** un paquet dans une interface, ce paquet se retrouve sur l'autre interface.*
  - ⋆ *une seule de ces deux interfaces recevra une configuration IP et correspondra au «* canal *», l'autre ne servira qu'à injecter des paquets vers la première.*

  ⟹ *On utilisera un adressage réseau privé pour les deux extrémités du «* tunnel *», ce qui permettra à la machine de router automatiquement le trafic vers ou depuis l'interface «* veth *» disposant d'une configuration IP.*
- ▷ le contenu de ces datagrammes doit être protégé :
  - ◇ on pourra faire de **l'obfuscation** par exemple, en inversant les bits du contenu du datagramme transporté avec un XOR et une suite de 1 de même taille ;
  - ◇ on pourra **chiffrer** avec AES le datagramme transporté en partageant au préalable une clé symétrique entre les interlocuteurs ;
- ▷ le protocole de communication dans lequel on va dissimuler notre canal caché doit :
  - ◇ disposer d'une « *charge utile* », c-à-d un contenu que peut choisir librement l'utilisateur, pour mettre le contenu de nos datagrammes transportés ;
  - ◇ avoir une autorisation de circulation entre les deux machines des interlocuteurs ;

  ⟹ les protocoles comme DNS et ICMP sont des bons candidats ;

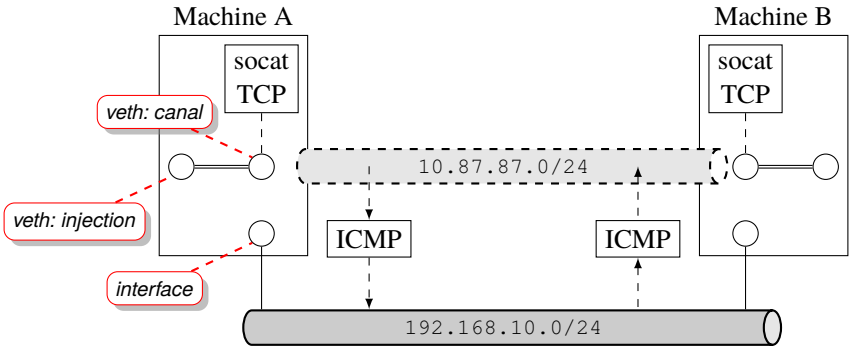
  ⟹ *On utilisera le protocole ICMP.*
- ▷ on utilisera Scapy pour :
  - ◇ l'interception des paquets à cacher depuis l'interface « *veth* » configurée en IP :

    Si on utilise `socat` comme outil :
    - ⋆ *on le fait communiquer entre les adresses des interlocuteurs situées dans le réseau du tunnel ;*
    - ⋆ *les paquets générés se retrouvent sur l'interface «* veth *» configurée en IP.*
  - ◇ l'encapsulation de ces paquets dans les paquets ICMP du tunnel (de la fragmentation peut être nécessaire si la taille des paquets dépassent celle de la charge utile d'ICMP) ;
  - ◇ la protection de ces paquets avec du chiffrement ou de l'obfuscation ;
  - ◇ l'envoi des paquets ICMP sur le réseau de la machine ;
  - ◇ l'interception des paquets ICMP sur l'interface réseau de la machine ;
  - ◇ l'extraction des paquets cachés à l'intérieur ;
  - ◇ l'injection de ces paquets sur l'interface « *veth* » non configurée en IP du récepteur.

    Si on utilise `socat` comme outil :
    - ⋆ *on le fait communiquer entre les adresses des interlocuteurs situées dans le réseau du tunnel ;*
    - ⋆ *les paquets injectés sur l'interface «* veth *» non configurée en IP vont être remis à l'outil `socat` ;*

    ⟹ *la communication se déroule correctement.*

Le schéma de fonctionnement du « *canal caché* »



Pour créer l'interface « *veth* » sur chaque interlocuteur :

xterm

```
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] cat creation_interface
#!/bin/bash

NETWORK=10.87.87
MACHINE=$1
INTERFACE_ENTREE=injection
INTERFACE_SORTIE=canal
sudo ip l add dev $INTERFACE_ENTREE type veth peer name $INTERFACE_SORTIE
sudo ip a add dev $INTERFACE_SORTIE $NETWORK.$MACHINE/30
sudo ip l set dev $INTERFACE_ENTREE up
sudo ip l set dev $INTERFACE_SORTIE up
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] ./creation_interface 2
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] ip address show dev canal
10: canal@injection: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether f6:0e:2f:e1:88:03 brd ff:ff:ff:ff:ff:ff
    inet 10.87.87.1/30 scope global canal
       valid_lft forever preferred_lft forever
    inet6 fe80::f40e:2fff:fee1:8803/64 scope link
       valid_lft forever preferred_lft forever
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] ip address show dev injection
11: injection@canal: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether ca:a9:10:da:58:0a brd ff:ff:ff:ff:ff:ff
    inet6 fe80::c8a9:10ff:feda:580a/64 scope link
       valid_lft forever preferred_lft forever
```

Pour intercepter avec Scapy le trafic sur deux interfaces simultanément :

xterm

```
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] cat tunnel_ecoute.py
#!/usr/bin/python3

from scapy.all import *

INTERFACE_LAN='h3-eth0'
INTERFACE_VETH='canal'

def traiter_trame(t):
    if UDP in t:
        print(t)
    if ICMP in t and (t[ICMP].type == 8):
        print(t)

sniff(iface=[INTERFACE_LAN,INTERFACE_VETH],prn=traiter_trame,filter='icmp or (host
10.87.87.2 and udp)')
```

Pour injecter un paquet dans l'interface `veth` non configurée appelée `injection` :

```
#!/usr/bin/python3

from scapy.all import *

INTERFACE_INJECTION='injection'
INTERFACE='canal'
ADRESSE_MAC=get_if_hwaddr(INTERFACE)

# Injection de paquets UDPs
sendp(Ether(src=RandMAC(),dst=ADRESSE_MAC)/IP(src='10.87.87.1',dst='10.87.87.2')/
   UDP(sport=5678,dport=6789)/"Hello",iface=INTERFACE_INJECTION)

# Injection de paquets TCPs
sendp(Ether(src=RandMAC(),dst=ADRESSE_MAC)/IP(src='10.87.87.1',dst='10.87.87.2')/
   TCP(sport=5678,dport=6789,flags='S'),iface=INTERFACE_INJECTION)
```

Pour tester l'injection et l'écoute avec UDP :

xterm

```
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] socat UDP-RECVFROM:6789 -
Hello
```

xterm

```
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] sudo python3 tunnel_ecoute.py
Ether / IP / UDP 10.87.87.1:5678 > 10.87.87.2:6789 / Raw
Ether / IP / UDP 10.87.87.1:5678 > 10.87.87.2:6789 / Raw
```

xterm

```
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] sudo python3 tunnel_injection.py
.
Sent 1 packets.
.
Sent 1 packets.
```

Pour TCP :

xterm

```
pef@hp:~/PROJET_AUDIT_SECRES$ [h3] socat tcp-listen:6789 -
```

xterm

```
Every 1.0s: sudo ss -tu state all      hp: Mon Apr 15 17:09:51 2024

Netid State    Recv-Q Send-Q Local Address:Port Peer Address:PortProcess
tcp   LISTEN   0      5            0.0.0.0:6789      0.0.0.0:*
tcp   SYN-RECV 0      0         10.87.87.2:6789   10.87.87.1:5678
```

Ici, on voit qu'on a bien reçu un segment `SYN`.

Pour le transport de TCP

Penser au fait que le connexions TCP ne sont pas faites par votre pile TCP/IP et qu'il va falloir configurer votre firewall NetFilter...

Pour le test et la réalisation du projet, vous utiliserez la plateforme habituelle, disponible sur « *git* » de l'Université (accessible uniquement par VPN) :

xterm

```
$ git clone https://git.p-fb.net/PeFClic/netlab.git
```

Le **travail** est :

- ▷ à réaliser en **binôme** ;
- ▷ à accompagner d'un **court rapport** contenant les choix et solutions de votre implémentation ainsi que des **captures d'écran** (échanges + tcpdump) de son fonctionnement ;
- ▷ à remettre sous forme **d'archive** par l'intermédiaire du service **filesender** de renater à `bonnefoi@unilim.fr`